**REMARKS**

Claims 1-3, 7, 9, 10, 12, 13, 20-22, 26, and 28-35 stand rejected under 35 U.S.C. 103(a) as unpatentable over US Publication 2004/0044997 by Talati (Talati) in view of alleged Applicants' admitted prior art (APA) and Mikhail Dmitriev "Safe Class and Data Evolution in Large and Long-Lived Data Applications," University of Glasgow, Scotland, UK, March 2001, pp. 1-191 (Dmitriev). Claims 11 and 23-25 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Talati in view of APA, Dmitriev, and US Patent 6,986,132 to Schwabe (Schwabe). Claims 14-17 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Talati, APA, and Dmitriev in view of US Patent 6,585,659 to Hiller (Hiller). Claims 18-19 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Talati in view of APA, Dmitriev, Hiller, and Schwabe.

Applicants thank the Examiner for the telephone interview of April 23, 2009. We discussed the present invention and proposed amendments. The Examiner suggested changing "memory" to "main memory" to which Applicants agreed. Applicants submit the proposed amendments and changes with this response.

Amendments to the claims

Applicants have amended claim 1 with the limitation "...a processor executing executable code stored on a main memory occupied by and used by an old code image and a temporary memory separate from the main memory~~storage device~~...." The amendment is well supported by the specification. See page 10, ¶ 35-36.

Claim 1 is further amended with the limitation "…a loader stored in the main memory and~~configured to~~ loading a new code image into the~~a~~ temporary memory…." The amendment is well supported by the specification. See page 10, ¶ 36; page 14, ¶ 53; fig. 404.

Applicants have further amended claim 1 with the limitation "…a branch module stored in the main memory causing the processor to execute a bootstrap module within the new code image…." The limitation is well supported by the specification. See page 15, ¶ 55; fig. 4, ref. 406.

Claim 1 is further amended with the limitation "…the bootstrap~~a logic~~ module ~~configured to~~ identifying incompatibilities between the old code image and the new code image from version information…." The amendment is fully supported by the specification. See fig. 4, showing the conversion module 412 within the bootstrap module 408, page 16, ¶ 60.

Applicants have also amended claim 1 with the limitation "…by accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information…" The amendment is well supported by the specification. See page 17, ¶ 62-65.

Claim 1 is further amended with the limitation "…associating persistent data of the old code image with the new code image, such that the persistent data is available in response to execution of a run-time segment of the new code image…" of claim 7. See also page 20, ¶ 75. In addition, Applicants have amended claim 1 with the limitation "…a copy module ~~configured to~~ copying the new code image into the main memory space occupied by the old code image …" to conform to other amendments. Claims 10, 13, 20, 29, and 30 are amended similar to claim 1.

Claim 14 is amended with the limitation "…the executable code further comprising a copy module stored within the new code image~~wherein the bootstrap code~~ overlaying~~s~~ the new code image in main memory with the old code image in response to reconciling~~ation of~~ the incompatibilities…." The amendment is well supported by the specification. See page 20, ¶ 76; fig. 4, ref. 414.

Applicants have amended claim 31 with the limitation "…the loader configures the temporary memory so that executable code is executed directly from the temporary memory...." The amendment is well supported by the specification. See page 14, ¶ 54.

Claim 31 is further amended with the limitations "…an update module stored in the memory maintaining an old code image pointer, a new code image pointer, capability fields storing the capability information, an old code image version number, a new code image version number, the old code image pointer, new code image pointer, capability fields, old code image version number, and new code image version number used by the bootstrap module…." The amendment is well supported by the specification. See page 15, ¶ 56.

Applicants have further amended claim 31 with the limitations "…wherein the bootstrap module follows the old code image pointer to locate an old code image header and a version field within the old code image header and follows the new code image pointer to locate a new code image header and a version field within the new code image header, the old code image header and the new code image header are organized according to the Microcode Reconstruct and Boot format...." The amendment is well supported by the specification. See page 16, ¶ 61.

Claim 31 is further amended with the limitations "…the bootstrap module reading the capability information from the old code image and the new code image and storing the capability information of the old code image and the new code image in the capability fields…." The amendment is well supported by the specification. See page 17, ¶ 62.

Applicants have further amended claim 31 with the limitation "...<u>the capability information comprising an indication that an EMULEX FLASH RAM is provided</u>...." The amendment is well supported by the specification. See page 18, ¶ 70.

Claim 31 is further amended with limitation "...<u>the persistent data comprising login tables</u>...." The amendment is well supported by the specification. See page 19, ¶ 75. Claims 32-35 are amended similar to claim 1. New claim 36 is also added with the limitations of claim 31.

Claims 1, 3, 10, and 11 are amended to remove "configured" language. Claims 2 and 21 are amended to remove "substantially." Claim 9 is amended to depend from claim 31. Claim 18 is amended to depend from claim 33. Claim 28 is amended to depend from claim 34. Claims 3, 11, 14, 15, 18, and 21-23 are amended to conform to amended predecessor claims. Claims 7, 19, and 24-26 are canceled.

<u>Response to rejections under 35 U.S.C. § 103</u>

Claims 1-3, 7, 9, 10, 12, 13, 20-22, 26, and 28-35 stand rejected under 35 U.S.C. 103(a) as unpatentable over Talati in view of APA and Dmitriev. Claims 11 and 23-25 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Talati in view of APA, Dmitriev, and Schwabe. Claims 14-17 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Talati, APA, and Dmitriev in view of Hiller. Claims 18-19 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Talati in view of APA, Dmitriev, Hiller, and Schwabe. Applicants respectfully traverse these rejections.

Claim 1 includes the limitations:

"…a processor executing executable code stored on a main memory occupied by and used by an old code image and a temporary memory separate from the main memory, the executable code comprising

a loader stored in the main memory and loading a new code image into the temporary memory;

a branch module stored in the main memory **causing the processor to execute a bootstrap module within the new code image**;

**the bootstrap module identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image, and by accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information, and reconciling the incompatibilities by changing an initialization order, converting a format of a data structure of the old code image to a format compatible with a data structure of the new code image, and associating persistent data of the old code image with the new code image, such that the persistent data is available in response to execution of a run-time segment of the new code image**; and

a copy module copying the new code image into the main memory space occupied by the old code image." Emphasis added.

Independent claims 10, 13, 20, 29, and 30 include similar limitations. Thus the present invention claims a processor executing executable code stored on a main memory occupied by and used by an old code image and a temporary memory separate from the main memory. See claim 1. In addition, the present invention claims loading a new code image into the temporary memory and causing the processor to execute a bootstrap module within the new code image. See claim 1. The bootstrap module identifies incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image, and by accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information. See claim 1. The bootstrap further reconciles the incompatibilities by changing an initialization order, converting a format of a data structure of the old code image to a format compatible with a data structure of the new code image, and associating persistent data of the old code image with the new code image, such that the persistent data is available in response to execution of a run-time segment of the new code image. See claim 1. The present invention further claims copying the new code image into the main memory space occupied by the old code image. See claim 1. Thus the present invention uses a bootstrap module within a new code image to identify and reconcile differences with an old code image before copying the new code image over the old code image. See page 20, ¶ 80-82.

Applicants submit that claim 1 is distinguished from Talati, APA, and Dmitriev by claiming "…execute a bootstrap module within the new code image…" and "…the bootstrap module identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image, and by accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information, and reconciling the incompatibilities by changing an initialization order, converting a format of a data structure of the old code image to a format compatible with a data structure of the new code image, and associating persistent data of the old code image with the new code image, such that the persistent data is available in response to execution of a run-time segment of the new code image…." Independent claims 10, 13, 20, 29, and 30 are similarly distinguished.

The Examiner notes that Talati teaches loading code into a shadow area separate from a runtime area and then copying the code to a runtime area. Office Action of March 18, 2009 (OA), page 3, line 18 – page 4, line 3. See also Talati, page 3, ¶ 37; page 4, ¶ 47; fig. 2, ref. 102, 108, 110, 218. The Examiner further notes that APA and Dmitriev teach indentifying incompatibilities and reconciling incompatibilities between old and new code images. OA, page 4, line 9 – page 6, line 11. However, Talati, APA, and Dmitriev do not disclose identifying and reconciling incompatibilities between old and new code images from code (a bootstrap module) within the new code image.

In Talati, the new version of executable (new code image) is staged to a shadow area and then copied to the runtime area without modification. See Talati, page 3, ¶ 38-39. Dmitriev teaches class promotion, including a static initialiser that modifies a legacy class when an object of the legacy class is loaded. Dmitriev, 2.1.3.3, page 21. APA discloses that new code images typically include changes in initialization processes, formats for data structures, and parameter lists. APA, page 2, ¶ 7. However, Talati, Dmitriev, and APA do not disclose executing a bootstrap module within the new code image that identifies and reconciles incompatibilities between the old code image and the new code image as we discussed. Applicants therefore submit that the combination of Talati, Dmitriev, and APA do not teach the elements of "…execute a bootstrap module within the new code image…" and "…the bootstrap module identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image, and by accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information,  and reconciling the incompatibilities by changing an initialization order, converting a format of a data structure of the old code image to a format compatible with a data structure of the new code image, and associating persistent data of the old code image with the new code image, such that the persistent data is available in response to execution of a run-time segment of the new code image…."

Applicants further submit that claim 1 is distinguished from Talati, APA, and Dmitriev by claiming "…identifying incompatibilities between the old code image and the new code image from … a difference in size and location between the old code image and the new code image…."  Independent claims 10, 13, 20, 29, and 30 are similarly distinguished. The Examiner notes that APA teaches that new code images may have changes in formats for data structures and parameters for interacting with other modules.  OA, page 4, lines 12-16, citing APA, page 2, ¶ 7.  However, APA does not disclose identifying incompatibilities between the old code image and the new code image from a difference in size and location between the old code image and the new code image.  Instead APA only states that differences exist.  Applicants therefore submit that APA, and also Talati and Dmitriev do not teach the element of "…identifying incompatibilities between the old code image and the new code image from … a difference in size and location between the old code image and the new code image…."

Because Talati, APA, and Dmitriev do not teach each element of independent claims 1, 10, 13, 20, 29, and 30, Applicants submit that claims 1, 10, 13, 20, 29, and 30 are allowable.

Furthermore, Applicants submit that dependent claims 2, 3, 9, 11, 12, 14-18, 21-23, 28, and 31-36 are allowable at least due to their dependency from independent claims 1, 10, 13, 20, 29, and 30.

## CONCLUSION

Applicants submit that the remarks and amendments put the present application in condition for allowance. In the event the Examiner finds any remaining impediment to the prompt allowance of any of these claims, which could be clarified in a telephone conference, the Examiner is respectfully urged to initiate the same with the undersigned.

Respectfully submitted,

/Brian C. Kunzler/

Brian C. Kunzler
Reg. No. 38,527
Attorney for Applicants

Date: May 18, 2009
Kunzler & McKenzie
8 East Broadway, Suite 600
Salt Lake City, UT 84111
Telephone (801) 994-4646
Fax (801) 531-1929